Projet économétrie des séries temporelles

Prédiction du taux d'investissement de Portugal

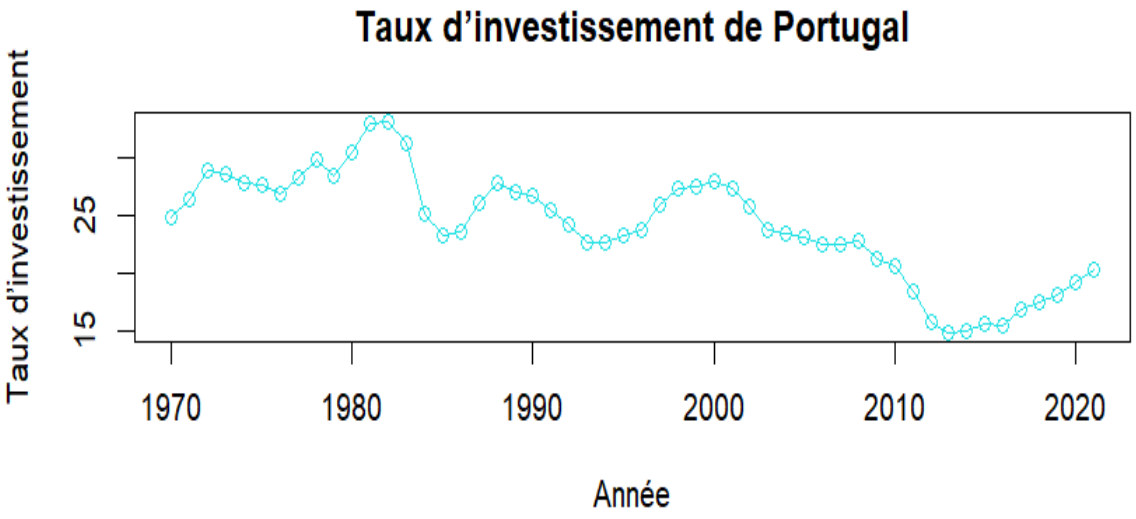
Yassine MOUDENE

I. Introduction

L'objectif de ce projet est de prédire le taux d'investissement de Portugal, pour faire nous allons utiliser des taux d'investissement annuels entre 1970 et 2021, ce qui donne 52 observations.

Ces données vont être utilisées pour estimer un modèle qui va être utilisé pour faire les prédictions, mais avant ça, on va passer par plusieurs étapes pour s'assurer que le modèle soit le plus précis possible. En premier, on analyse le chronogramme de la série et puis il faut faire des tests pour savoir si la série est stationnaire, sinon on la rendre stationnaire, et après, on essaye plusieurs modèles et choisir le meilleur pour qu'on puisse à la fin faire la prédiction.

II. Chronogramme



On observe une tendance décroissante sur l'ensemble des données. Par contre, on peut dire qu'il n'y a pas de saisonnalité, parce que les données sont annuelles, mais on ne peut pas dire si la série est stationnaire ou non, il faut faire des tests pour vérifier ça. Cependant, il y a une baisse importante après 2010, ce qui peut être expliqué par la crise économique connue par le pays qui a duré plusieurs années. Le pays avait des niveaux élevés de dette publique et luttait pour maintenir la croissance économique. En mai 2010, le Portugal a reçu un plan de sauvetage de 78 milliards d'euros du Fonds monétaire international (FMI), de la Banque centrale européenne (BCE) et de

la Commission européenne (CE). Ce renflouement était assorti de conditions strictes, notamment des mesures d'austérité telles que la réduction des dépenses publiques et l'augmentation des impôts.

La réduction des dépenses publiques et l'augmentation des impôts peuvent également entraîner une diminution des investissements publics dans les infrastructures et autres biens publics, ce qui peut rendre le pays moins attrayant pour les investisseurs.

III. Stationnarité

En général, la série doit être stationnaire avant la modélisation lorsque l'on utilise des modèles de séries chronologiques, tels que les modèles ARIMA (Autoregressive Integrated Moving Average) pour faire des prévisions.

Une série stationnaire est une série qui ne présente pas de tendance ou de saisonnalité systématique et dont les propriétés statistiques telles que la moyenne et la variance sont constantes dans le temps. Si une série n'est pas stationnaire, cela peut fausser les prévisions et rendre les résultats moins fiables.

Ainsi, avant de modéliser une série, il est important de vérifier si elle est stationnaire ou non. Si elle ne l'est pas, il peut être nécessaire d'appliquer des transformations telles que la différenciation ou la décomposition saisonnière pour rendre la série stationnaire avant de procéder à la modélisation. Pour cela nous allons utiliser les tests de racine unitaire qui vont permettre de déterminer si le PDG est stationnaire, TS (Trend-Stationary) ou DS (Difference-Stationary).

1. Test de Dickey Fuller

Le test de Dickey-Fuller (DF) est un test statistique utilisé pour déterminer si une série chronologique est stationnaire ou non. Le test de Dickey-Fuller est basé sur un modèle autorégressif (AR) d'ordre 1. Il teste l'hypothèse nulle selon laquelle une série chronologique possède une racine unitaire (pas stationnaire) contre l'hypothèse alternative selon laquelle la série ne possède pas de racine unitaire (stationnaire).

Pour DF, il existe 3 spécifications possibles : "trend", "drift" ou "none".

Avec la spécification "trend" nous avons l'équation suivante:

$$\triangle Xt = (\rho - 1)Xt{-}1 + \beta 0 + \beta 1 tendancet + \epsilon t$$

On teste les hypothèses suivantes :

$$H0 : \rho - 1 = 0 \text{ et } \beta 1 = 0$$

$$Ha : | \rho | < 1 \text{ et } \beta 1 \neq 0$$

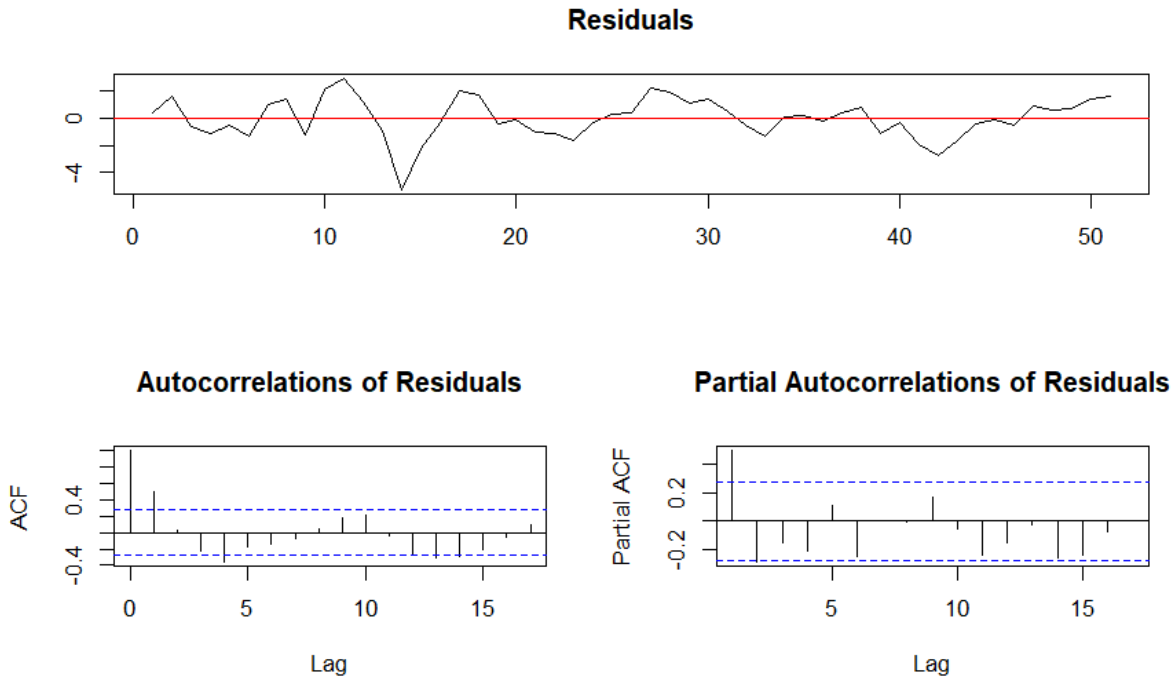
Si H0 est vraie alors le PGD est DS et si Ha est vraie alors le PGD est TS.

```
> summary(ur.df(PRT_inv,type="trend",lag=0))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt)

Residuals:
    Min      1Q  Median      3Q     Max
-5.2466 -0.9795 -0.0878  1.0654  2.8964

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.08935    2.38675   2.551   0.0140 *
z.lag.1     -0.19353    0.07596  -2.548   0.0141 *
tt          -0.05705    0.02412  -2.366   0.0221 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.499 on 48 degrees of freedom
Multiple R-squared:  0.1236,    Adjusted R-squared:  0.08706
F-statistic: 3.384 on 2 and 48 DF,  p-value: 0.04218


Value of test-statistic is: -2.5478 2.3159 3.384

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -4.04 -3.45 -3.15
phi2  6.50  4.88  4.16
phi3  8.73  6.49  5.47
```

On trouve que le coefficient β1 est significatif, vue que sa p-value 0.0221 est inférieure à 0.05. La valeur de la statistique t pour (ρ – 1) est de -2,5478, ce qui est supérieur à la valeur critique au niveau de 5% (-3,45). Cela signifie que nous ne pouvons pas

rejeter l'hypothèse nulle du test et conclure que la série est non stationnaire. On a ρ – 1 = 0, alors notre série est DS.

Cependant, la validité du test de Dickey Fuller dépend des résidus, donc le test n'est valable que si les résidus ne sont pas autocorrélés. On teste cela avec l'ACF et le PACF.



Les résidus sont autocorrélés, donc il faut passer au test de Dickey Fuller Augmenté (ADF).

2. Test de Dickey Fuller Augmenté

Le test de Dickey-Fuller augmenté est une version améliorée du test de Dickey-Fuller standard, qui inclut des termes supplémentaires pour prendre en compte les effets de la tendance et de la saisonnalité. L'équation devient :

$$\Delta X_t = (\rho - 1)\, X_{t-1} + \beta_0 + \beta_1 t + \epsilon_t + \sum \gamma_p \Delta X_{t-p}$$

Afin de déterminer le nombre de p retards qu'il faut ajouter, On va utiliser le critère MAIC.

```
> summary(CADFtest(PRT_inv ,criterion="MAIC", type="trend", max.lag.y=pmax))
Augmented DF test

                                              ADF test
t-test statistic:                          -2.2699250
p-value:                                    0.4401283
Max lag of the diff. dependent variable:    3.0000000

Call:
dynlm(formula = formula(model), start = obs.1, end = obs.T)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5275 -0.4726 -0.0303  0.7740  1.8559

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.91421    3.36526   2.055   0.0474 *
trnd        -0.06050    0.03562  -1.698   0.0983 .
L(y, 1)     -0.21871    0.09635  -2.270   0.4401
L(d(y), 1)   0.73892    0.13710   5.389 4.93e-06 ***
L(d(y), 2)  -0.14316    0.16465  -0.869   0.3905
L(d(y), 3)  -0.05053    0.14691  -0.344   0.7329
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.131 on 35 degrees of freedom
Multiple R-squared:  0.5479,    Adjusted R-squared:  0.4833
F-statistic: 12.15 on 3 and 35 DF,  p-value: 1.327e-05
```

On obtient p =3 après le MAIC, donc on peut utiliser ce nombre de lag pour le test ADF.

```
> summary(ur.df(PRT_inv,type="trend",lag=3))

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7761 -0.6701  0.0524  0.6616  3.1189

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.70901    2.73044   2.823  0.00724 **
z.lag.1     -0.24441    0.08494  -2.877  0.00628 **
tt          -0.06828    0.02652  -2.574  0.01365 *
z.diff.lag1  0.64247    0.14082   4.562 4.35e-05 ***
z.diff.lag2 -0.08280    0.15774  -0.525  0.60239
z.diff.lag3 -0.01113    0.14450  -0.077  0.93898
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.242 on 42 degrees of freedom
Multiple R-squared:  0.4265,    Adjusted R-squared:  0.3582
F-statistic: 6.246 on 5 and 42 DF,  p-value: 0.0002046


Value of test-statistic is: -2.8773 2.9004 4.1505

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -4.04 -3.45 -3.15
phi2  6.50  4.88  4.16
phi3  8.73  6.49  5.47
```

On accepte H0 : ρ−1 = 0 car la statistique t (-2.8773) est plus grande que -1.95, On a donc un processus DS avec présence de racine unitaire.

3. Test de Zivot et Andrews

Le test de Zivot et Andrews est souvent utilisé pour tester si une série temporelle a une rupture structurelle dans le contexte de la modélisation des données macroéconomiques, ce qui n'est pas pris par les tests de DF et d'ADF. Le test est utile pour détecter des changements significatifs dans le comportement d'une série, tels que des changements dans la tendance, la volatilité ou la corrélation, qui peuvent être causés par des événements économiques tels que des crises financières, des récessions ou des changements de politique.

H0 : DS sans changement structurel, ρ = 1

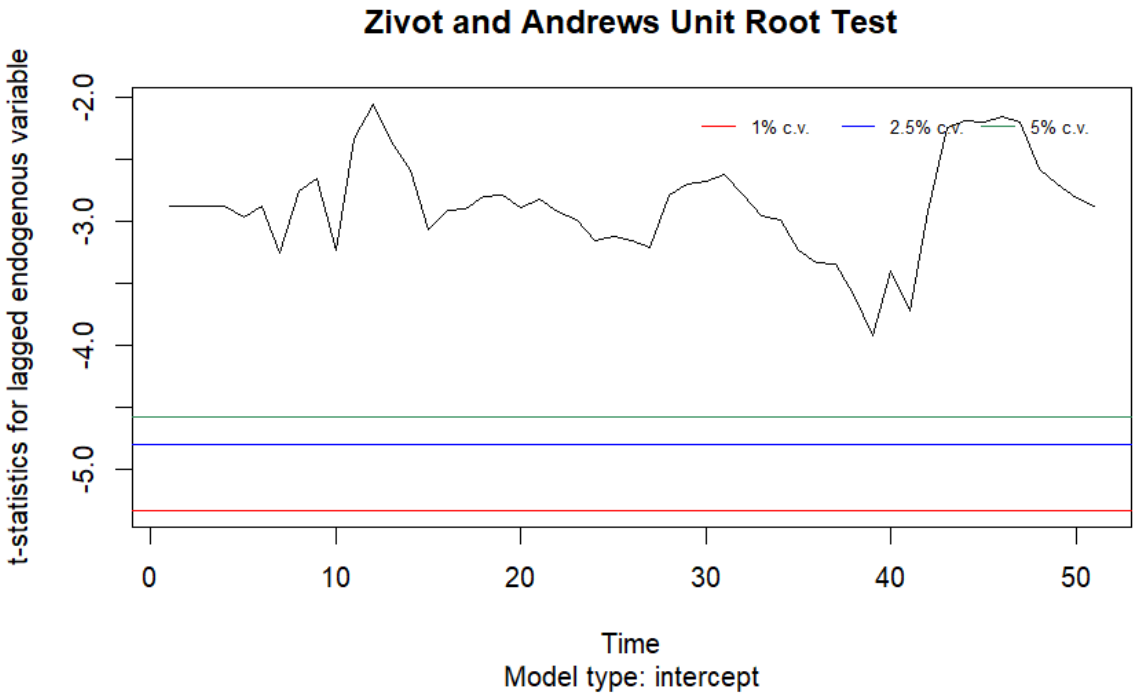
Ha : TS avec un unique changement structurel, | ρ |< 1

On utilise le nombre de lag donné par MAIC.

```
> summary(ur.za(PRT_inv, model="intercept", lag=3))

################################
# Zivot-Andrews Unit Root Test #
################################


Call:
lm(formula = testmat)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3500 -0.5996 -0.1000  0.5721  3.1211

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.672578   3.039624   3.840 0.000418 ***
y.l1         0.609164   0.099639   6.114 2.98e-07 ***
trend       -0.062517   0.025144  -2.486 0.017070 *
y.dl1        0.672550   0.133493   5.038 9.92e-06 ***
y.dl2       -0.005502   0.152146  -0.036 0.971326
y.dl3        0.074109   0.140689   0.527 0.601203
du          -1.837601   0.742232  -2.476 0.017516 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.173 on 41 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.9476,    Adjusted R-squared:   0.94
F-statistic: 123.7 on 6 and 41 DF,  p-value: < 2.2e-16


Teststatistic: -3.9225
Critical values: 0.01= -5.34 0.05= -4.8 0.1= -4.58

Potential break point at position: 39
```

Pour la spécification” both”, on a trouvé que δ2 n’est pas significatif 0.05494 > 0.05, alors on a employé la spécification ”intercept”. On trouve que δ1 a une p-value de 0.017516 < 0.05, donc la spécification "intercept" est correcte. La statistique calculée -3.9225 > la valeur critique à 5% =-4.8, donc on accepte H0. On a DS sans changement structurel.

La date de rupture est à la 39ème observation soit 2008.



4. Test de Lee et Strazicich

On va utiliser le test de Lee et Strazicich parce qu’il a la possibilité de deux dates de rupture, ce qui n’est pas le cas pour le test de Zivot et Andrews. Comme on avait seulement δ1 significatif dans ZA, alors on choisit la spécification "crash", ce qui donne les hypothèses suivantes :

$$\text{H0} : yt = \mu 0 + d1B1t + d2B2t + yt{-}1 + v1t$$

$$\text{H1} : yt = \mu 1 + \gamma * trendt + d1D1t + d2D2t + v2t$$

```
> myBreaks <- 1
> myModel <- "crash"
> myLags <- 3
> myLS_test <- ur.ls(y= PRT_inv , model = myModel, breaks = myBreaks, lags = myLags, method = "GTOS",pn = 0.1, print.results = "print" )
[1] -2.850924
[1] "First possible structural break at position: 7"
[1] "The location of the first break - lambda_1: 0.1 , with the number of total observations: 52"
Critical values - Crash model:
       1%    5%   10%
[1,] -4.239 -3.566 -3.211
[1] "Number of lags determined by general-to-specific lag selection: 1"
```

λ est estimé à 0.1 et Tb = 7 donc 1976 est la date de rupture. Comme La valeur de la t statistique calculée est -2.850924 > -3.566, on accepte H0 donc le PGD est DS sans changement structurel. On a aussi essayé le test avec deux breaks et on arrive à la même conclusion.

5. Bootstrap Lee et Strazicich

Vue qu'on a un petit échantillon de 52 observations, on va utiliser ls.bootstrap, ce qui donne le résultat suivant :

```
> myBreaks <- 1
> myModel <- "crash"
> myParallel_LS <- ur.ls.bootstrap(y = PRT_inv , model = myModel, breaks = myBreaks, lags = myLags,
"Fixed",pn = 0.1, critval = "bootstrap", print.results = "print")
[[1]]
[1] -1.928239

[1] "First possible structural break at position: 15"
[1] "The location of the first break - lambda_1: 0.3 , with the number of total observations: 52"
Critical values - Crash model:
         1%    5%   10%
[1,] -4.239 -3.566 -3.211
[1] "Number of lags used: 3"
```

λ est estimé à 0.3 et Tb = 15 donc 1984 est la date de rupture. Comme La valeur de la t statistique calculée est -1.928239 > -3.566, on accepte H0 donc le PGD est DS sans changement structurel. On arrive à la même conclusion avec le test avec deux breaks.

6. Differentiation:

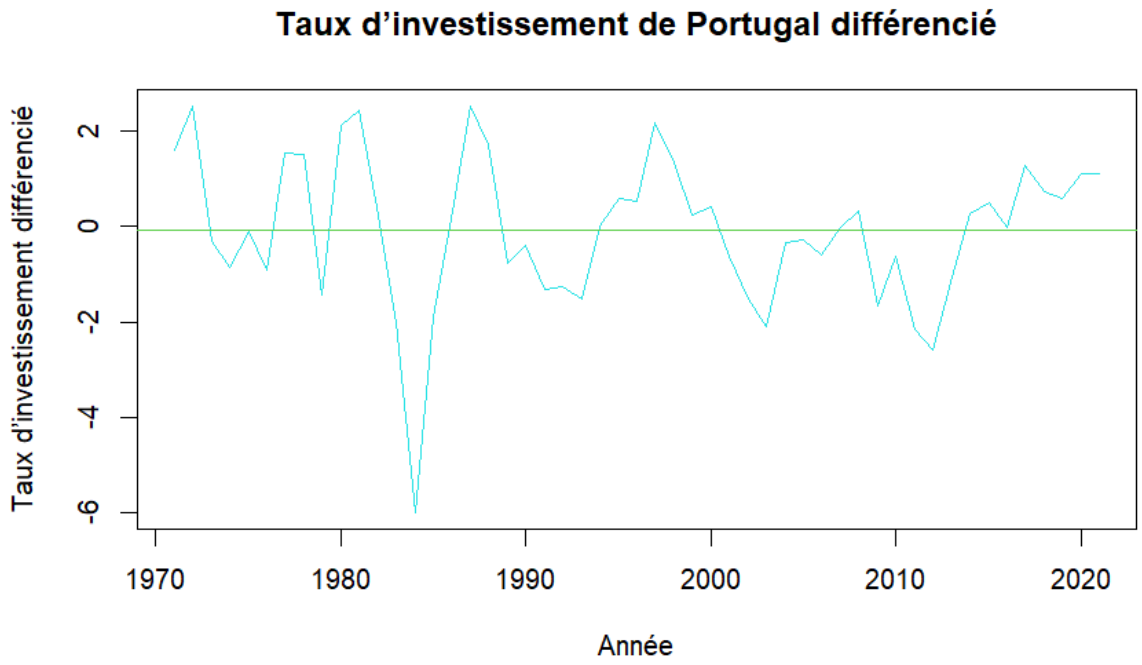
On a un processus DS qui s'écrit comme :

$$Xt = \delta + Xt-1 + ut$$

Avec **ut** un BB et δ une constante appelée la dérive.

Cette série n'est pas stationnaire, donc on va utiliser la différentiation pour la rendre stationnaire. On va utiliser la fonction *diff ()* de R.

Voilà la nouvelle série différenciée :

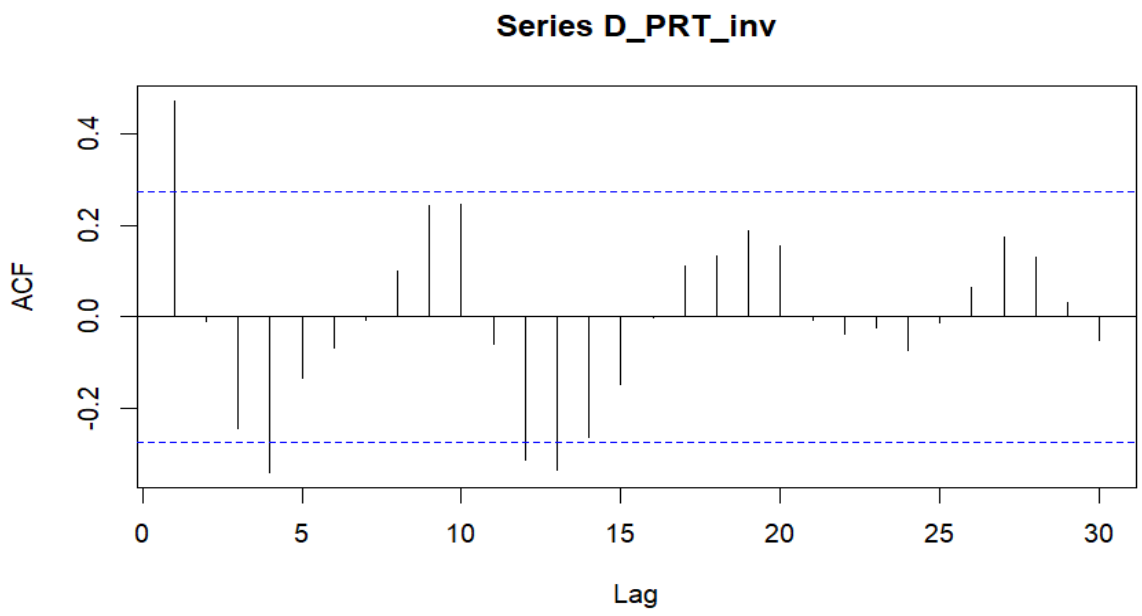


La série différenciée présente des propriétés de stationnarité, comme la variation autour de la moyenne, dans ce cas 0, et une variance plus ou moins constante. Donc, on peut passer à la modélisation.

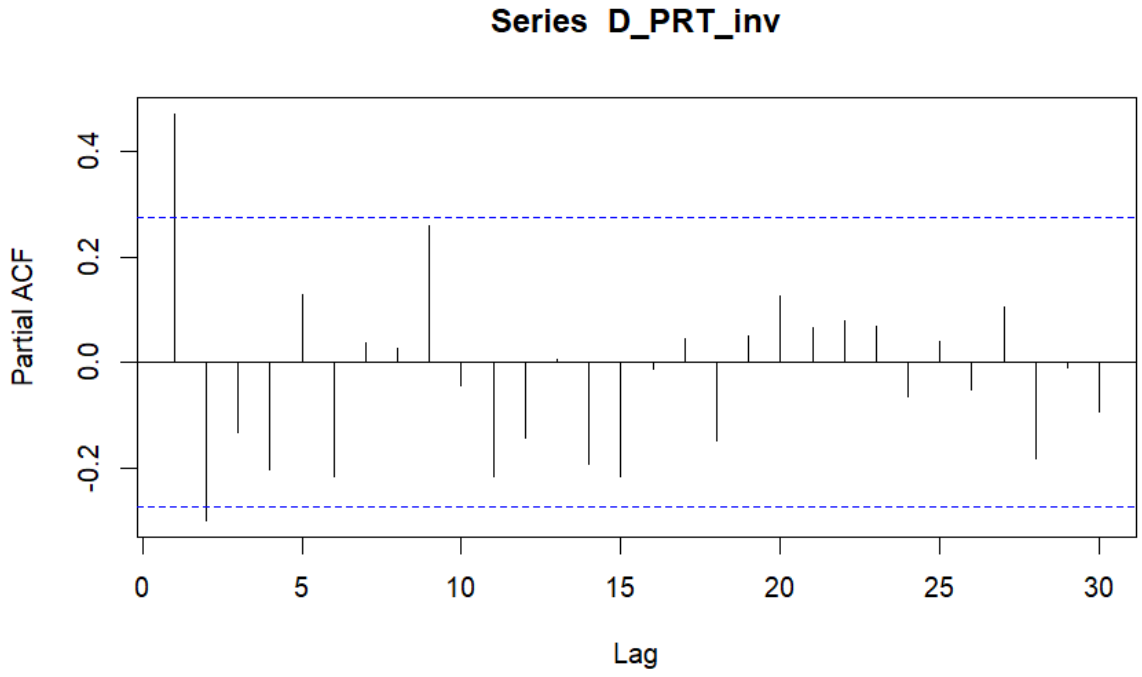
IV. Modélisation

Maintenant qu'on a une série stationnaire on peut passer à la modélisation. En premier on affiche les graphes d'ACF et de PACF pour voir s'il y a de l'autocorrélation et un test de Ljung-box pour confirmer qu'il y a de l'autocorrélation et puis on utilise la fonction *eacf()* pour obtenir les valeurs de p et q du modèle et on va simuler plusieurs modèles afin de choisir le meilleur.

- ACP et PACF



On peut voir qu'il y a de l'autocorrélation jusqu'à l'ordre 13. Les lags significatifs sont 1, 4, 12 et 13. On constate aussi que les coefficients de corrélation se dégradent progressivement, on peut dire qu'il s'agit probablement d'un processus AR.



Pour la PACF il y a de l'autocorrélation jusqu'à l'ordre 2. Il y a que deux lags significatifs 1 et 2.

On fait un test de Ljung-box pour confirmer l'existence d'autocorrélation.

Hypothèse nulle : Il n'y a pas d'autocorrélation dans les données.

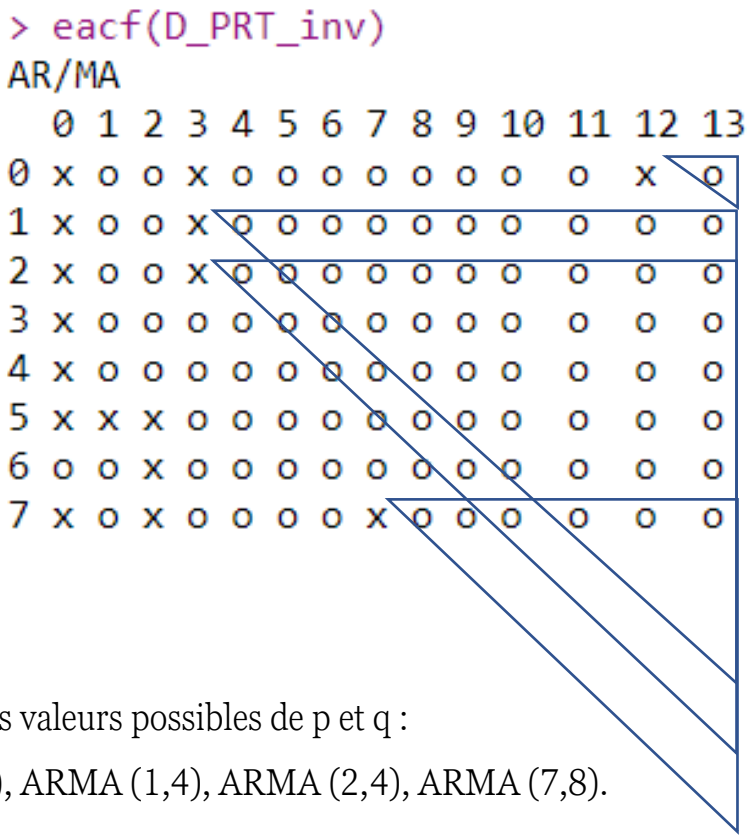
Hypothèse alternative : Il y a de l'autocorrélation dans les données.

```
> Box.test(D_PRT_inv, type = "Ljung-Box", lag = 13)

        Box-Ljung test

data:  D_PRT_inv
X-squared = 46.836, df = 13, p-value = 1.03e-05
```

La p-value est inférieure à 0.05, on rejette H0, ce qui confirme qui st qu'il y a de l'autocorrélation.

- **EACF**

On utilise l'EACF pour déterminer l'ordre de p et q. L'EACF nous donne le résultat suivant :



```
> eacf(D_PRT_inv)
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x o o x o o o o o o o  o  x  o
1 x o o x o o o o o o o  o  o  o
2 x o o x o o o o o o o  o  o  o
3 x o o o o o o o o o o  o  o  o
4 x o o o o o o o o o o  o  o  o
5 x x x o o o o o o o o  o  o  o
6 o o x o o o o o o o o  o  o  o
7 x o x o o o o x o o o  o  o  o
```

Il y a plusieurs valeurs possibles de p et q :

ARMA (0,13), ARMA (1,4), ARMA (2,4), ARMA (7,8).

On obtient les modèles suivants avec des coefficient significatifs :

- $Yt = \theta 1 \epsilon t-1 + \epsilon t$ (1) : BIC = 180.6155
- $Yt = \varphi 0 + \varphi 1 Xt-1 + \theta 3 \epsilon t-3 + \theta 4 \epsilon t-4 + \epsilon t$ (2) : BIC = 184.5212
- $Yt = \varphi 1 Xt-1 + \theta 2 \epsilon t-2 + \epsilon t$ (3) : BIC = 183.8448

On choisit celui avec le plus petit BIC, dans ce cas, le modèle 1 ARIMA(0,0,1) :

$$Yt = \theta 1 \epsilon t-1 + \epsilon t$$

```
z test of coefficients:

    Estimate Std. Error z value  Pr(>|z|)
ma1  0.64990    0.14077  4.6169 3.895e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Tests d’autocorrélation sur les résidus

Maintenant que nous avons le modèle avec des coefficients significatifs, on doit s’assurer que les aléas du modèle sont des Bruit Blanc.

a. T test

H0 : l’espérance des aléas est nulle

H1 : l’espérance des aléas n’est pas nulle

```
> t.test(reg1$residuals)

        One Sample t-test

data:  reg1$residuals
t = -0.27253, df = 50, p-value = 0.7863
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.4219895  0.3211567
sample estimates:
  mean of x
-0.05041642
```

La p-value (0.7863) est supérieure à 5%, on accepte H0, donc l’espérance des aléas est nulle.

b. ARCH test

H0 : Pas d'effet d'Arch

H1 : L'existence d'effet d'Arch

```
> ArchTest(reg1$residuals, lags = 13)

        ARCH LM-test; Null hypothesis: no ARCH effects

data:  reg1$residuals
Chi-squared = 6.8105, df = 13, p-value = 0.9117
```

La p-value (0.9117) est supérieure à 5%, on accepte H0, donc il n'y a pas d'effet d'Arch.

c. Ljung Box test

H0 : Il n'y a pas d'autocorrélation.

H1 : Il y a de l'autocorrélation.

```
> Box.test(reg1$residuals, type="Ljung-Box", lag=13)

        Box-Ljung test

data:  reg1$residuals
X-squared = 17.126, df = 13, p-value = 0.1936
```

La p-value (0.1936) est supérieure à 5%, on accepte H0, donc il n'y a pas d'autocorrélation.

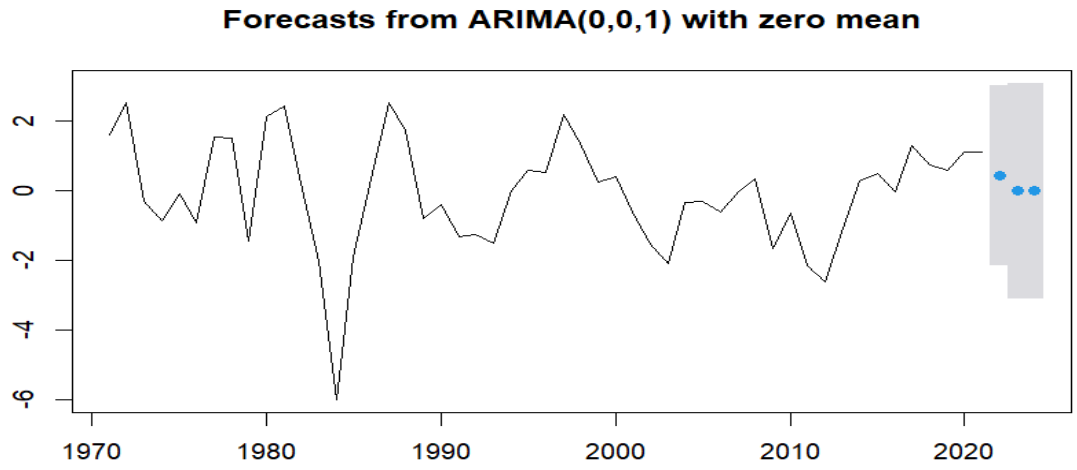
Enfin, on peut conclure que les aléas sont des bruits blancs. Donc, on peut passer à la prévision.

V. Prévision

On va utiliser le modèle suivant pour faire la prévision pour les cinq années prochaines :

ARIMA(0,0,1) : $Yt = \theta 1 \epsilon t-1 + \epsilon t$

On fait la prévision sur la série différenciée.

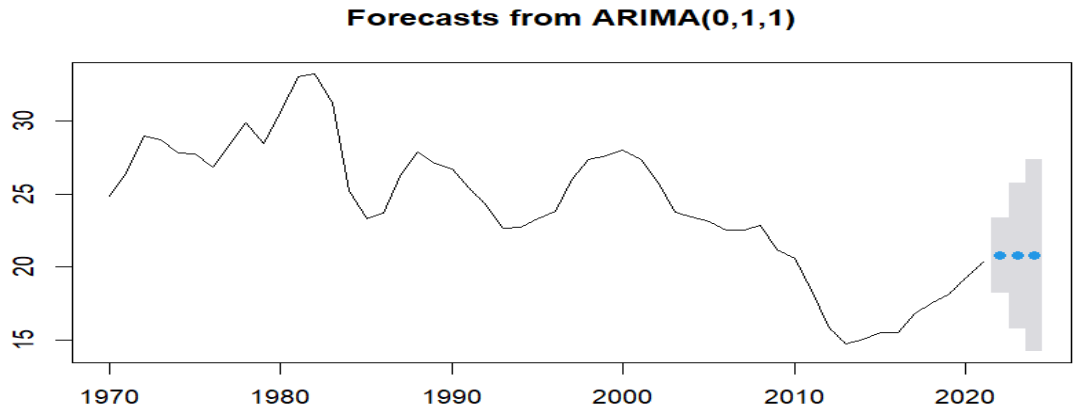


On obtient les valeurs suivantes :

	Point Forecast <dbl>	Lo 95 <dbl>	Hi 95 <dbl>
2022	0.4444667	-2.146818	3.035751
2023	0.0000000	-3.090454	3.090454
2024	0.0000000	-3.090454	3.090454

Un processus MA ne retient pas beaucoup d'information, c'est pourquoi il converge à la moyenne à des ordres supérieures, donc on ne peut prédire qu'à l'ordre 1 à chaque fois.

Prévision sur la série sur la série originale.



	Point Forecast <dbl>	Lo 95 <dbl>	Hi 95 <dbl>
2022	20.76447	18.17317	23.35576
2023	20.76447	15.76509	25.76384
2024	20.76447	14.18627	27.34267

Annexe

2023-04-25

Chargement des library.

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(urca)
library(forecast)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ─
─
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ─
─
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all confl
icts to become errors
```

```
library(TSstudio)
library(lmtest)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(TSA)
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
##
```

```
## Attaching package: 'TSA'
##
## The following object is masked from 'package:readr':
##
##     spec
##
## The following objects are masked from 'package:stats':
##
##     acf, arima
##
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(CADFtest)
```

```
## Loading required package: dynlm
## Loading required package: sandwich
## Registered S3 methods overwritten by 'CADFtest':
##   method     from
##   bread.mlm  sandwich
##   estfun.mlm sandwich
```

```
library(caschrono)
library(FinTS)
```

```
##
## Attaching package: 'FinTS'
##
## The following object is masked from 'package:forecast':
##
##     Acf
```

```
library(doSNOW)
```

```
## Loading required package: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## Loading required package: iterators
## Loading required package: snow
```

```
library(parallel)
```

```
##
## Attaching package: 'parallel'
##
```

```
## The following objects are masked from 'package:snow':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, clusterSplit, makeCluster, parApply,
##     parCapply, parLapply, parRapply, parSapply, splitIndices,
##     stopCluster
```

Chargement des données.

```
Portugal = Portugal <- read_csv("C:/Users/moude/Desktop/Portugal.csv")
```

```
## Rows: 52 Columns: 4
## ── Column specification ────────────────────────────────────────────────────
─
## Delimiter: ","
## dbl (4): TIME, Value, PIB, Taux
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this messag
e.
```

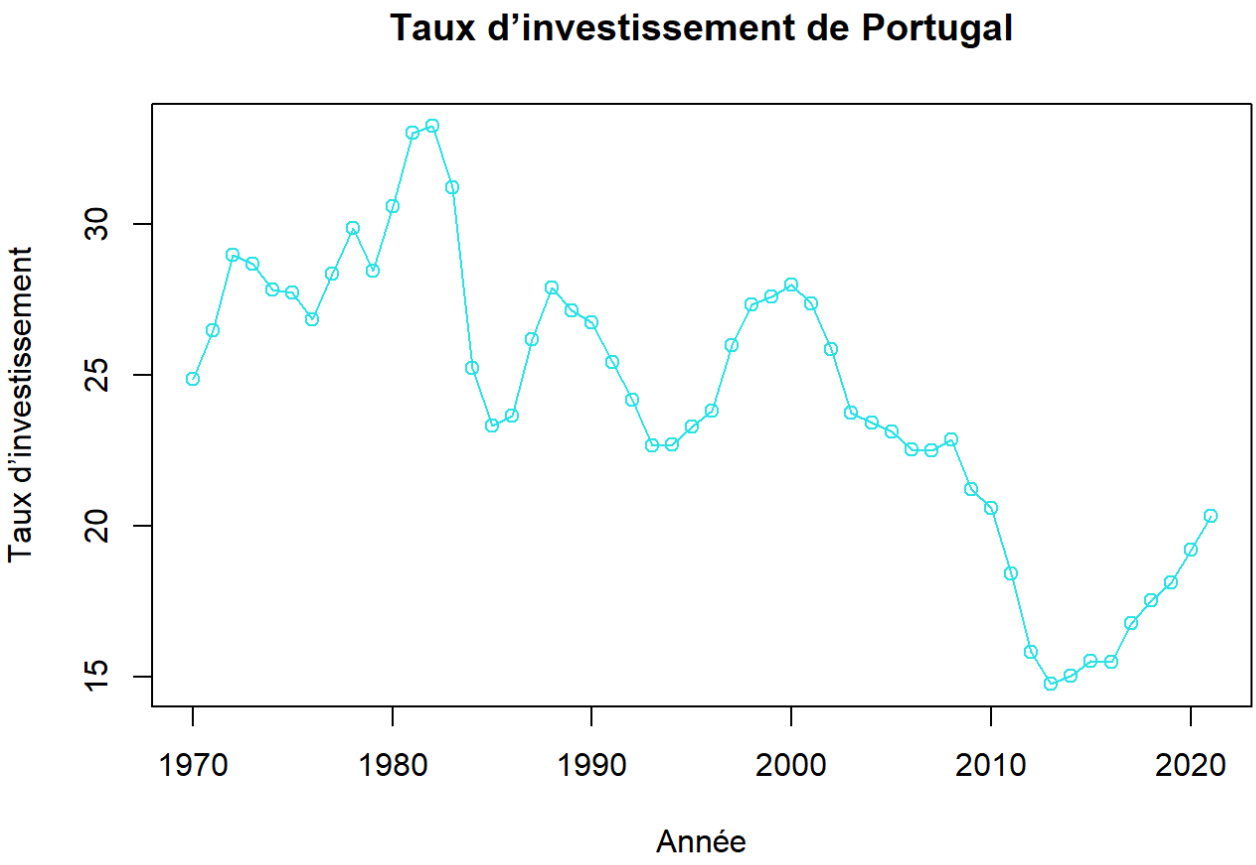
```
PRT_inv = Portugal[,"Taux"]
```

La série temporelle.

```
PRT_inv = ts(PRT_inv,freq=1, start = 1970)
```

Chronogramme

```
plot.ts(PRT_inv,xlab="Année", ylab="Taux d'investissement", main="Taux d'investiss
ement de Portugal",type='o',col=5)
```



On observe une tendance décroissante sur l'ensemble des données. Par contre, on peut dire qu'il n'y a pas de saisonnalité, parce que les données sont annuelles, mais on ne peut pas dire si la série est stationnaire ou non, il faut faire des tests pour vérifier ça. Cependant, il y a une baisse importante après 2010, ce qui peut être expliqué par la crise économique connue par le pays qui a duré plusieurs années.

Tests de racine unitaire

Test de Dickey Fuller

On commence par l specification "trend"

```
summary(ur.df(PRT_inv,type="trend",lag=0))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2466 -0.9795 -0.0878  1.0654  2.8964
```

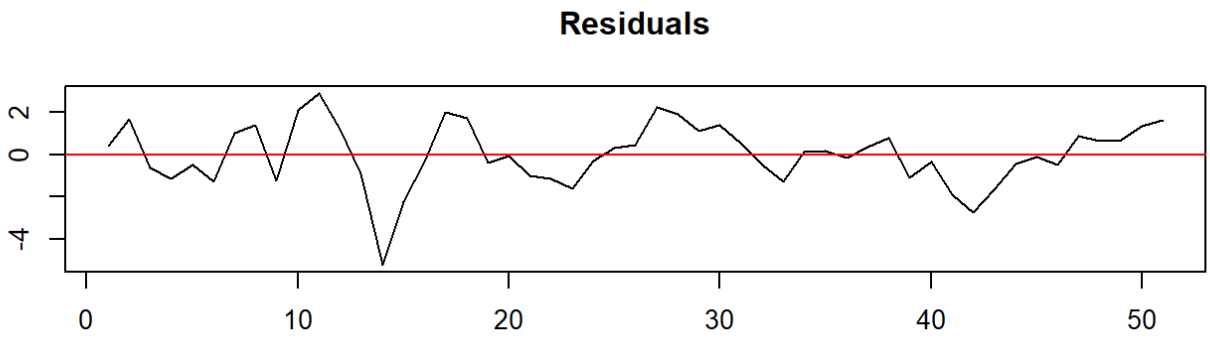
```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.08935    2.38675   2.551   0.0140 *
## z.lag.1     -0.19353    0.07596  -2.548   0.0141 *
## tt          -0.05705    0.02412  -2.366   0.0221 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.499 on 48 degrees of freedom
## Multiple R-squared:  0.1236, Adjusted R-squared:  0.08706
## F-statistic: 3.384 on 2 and 48 DF,  p-value: 0.04218
##
##
## Value of test-statistic is: -2.5478 2.3159 3.384
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

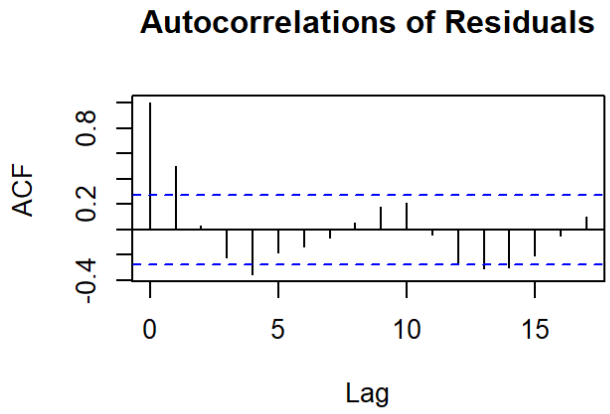
Le beta_1 (tt) est significatif et t calculé pour (ρ − 1) est superieure à la valeur critique (-2.5478 > -3.45), alors on accepte H0 alors on a un processus DS.

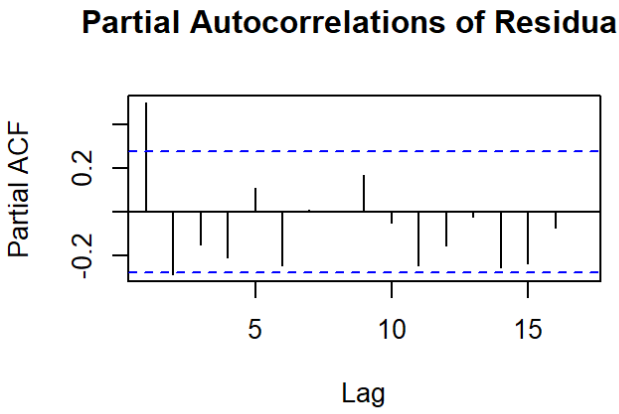
L'autocorrélation des résidus

Ces résultats ne sont valables que si les résidus ne sont pas autocorrélés.

```
plot(ur.df(PRT_inv,type="trend",lag=0))
```







On constate qu'il y a de l'autocorrélation entre les résidus, donc on passe à L'ADF. Mais, il faut d'abord trouver le nombre de lag à utiliser pour le test d'ADF. On va utiliser le critère MAIC pour les determiner.

Test de Dickey Fuller Augmenté

Le nombre maximum des lags.

```
Tinv = length(PRT_inv)
pmax<-as.integer(12*(Tinv/100)^(0.25))
print(pmax)
```

```
## [1] 10
```

MAIC

```
summary(CADFtest(PRT_inv ,criterion="MAIC", type="trend", max.lag.y=pmax))
```

```
## Augmented DF test
##                                                ADF test
## t-test statistic:                           -2.2699250
## p-value:                                     0.4401283
## Max lag of the diff. dependent variable:     3.0000000
##
## Call:
## dynlm(formula = formula(model), start = obs.1, end = obs.T)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.5275 -0.4726 -0.0303  0.7740  1.8559
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.91421    3.36526   2.055   0.0474 *
## trnd        -0.06050    0.03562  -1.698   0.0983 .
## L(y, 1)     -0.21871    0.09635  -2.270   0.4401
## L(d(y), 1)   0.73892    0.13710   5.389 4.93e-06 ***
## L(d(y), 2)  -0.14316    0.16465  -0.869   0.3905
## L(d(y), 3)  -0.05053    0.14691  -0.344   0.7329
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.131 on 35 degrees of freedom
## Multiple R-squared:  0.5479, Adjusted R-squared:  0.4833
## F-statistic: 12.15 on 3 and 35 DF,  p-value: 1.327e-05
```

Max lag of the diff. dependent variable est égale à 3, donc on va utiliser un nombre de lags égale à 3 pour le test de l'ADF.

Test de Dickey Fuller Augmenté avec p = 3.

```
summary(ur.df(PRT_inv,type="trend",lag=3))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7761 -0.6701  0.0524  0.6616  3.1189
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.70901    2.73044   2.823  0.00724 **
## z.lag.1     -0.24441    0.08494  -2.877  0.00628 **
## tt          -0.06828    0.02652  -2.574  0.01365 *
## z.diff.lag1  0.64247    0.14082   4.562 4.35e-05 ***
## z.diff.lag2 -0.08280    0.15774  -0.525  0.60239
## z.diff.lag3 -0.01113    0.14450  -0.077  0.93898
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.242 on 42 degrees of freedom
## Multiple R-squared:  0.4265, Adjusted R-squared:  0.3582
## F-statistic: 6.246 on 5 and 42 DF,  p-value: 0.0002046
##
##
```

```
## Value of test-statistic is: -2.8773 2.9004 4.1505
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

On accepte H0 : ρ−1 = 0 car la statistique t (-2.8773) est plus grande que -1.95, On a donc un processus DS avec présence de racine unitaire.

Test de Zivot et Andrews

Le test de Zivot et Andrews est souvent utilisé pour tester si une série temporelle a une rupture structurelle dans le contexte de la modélisation des données macroéconomiques, ce qui n'est pas pris par les tests de DF et d'ADF.

```
summary(ur.za(PRT_inv, model="both", lag=3))
```

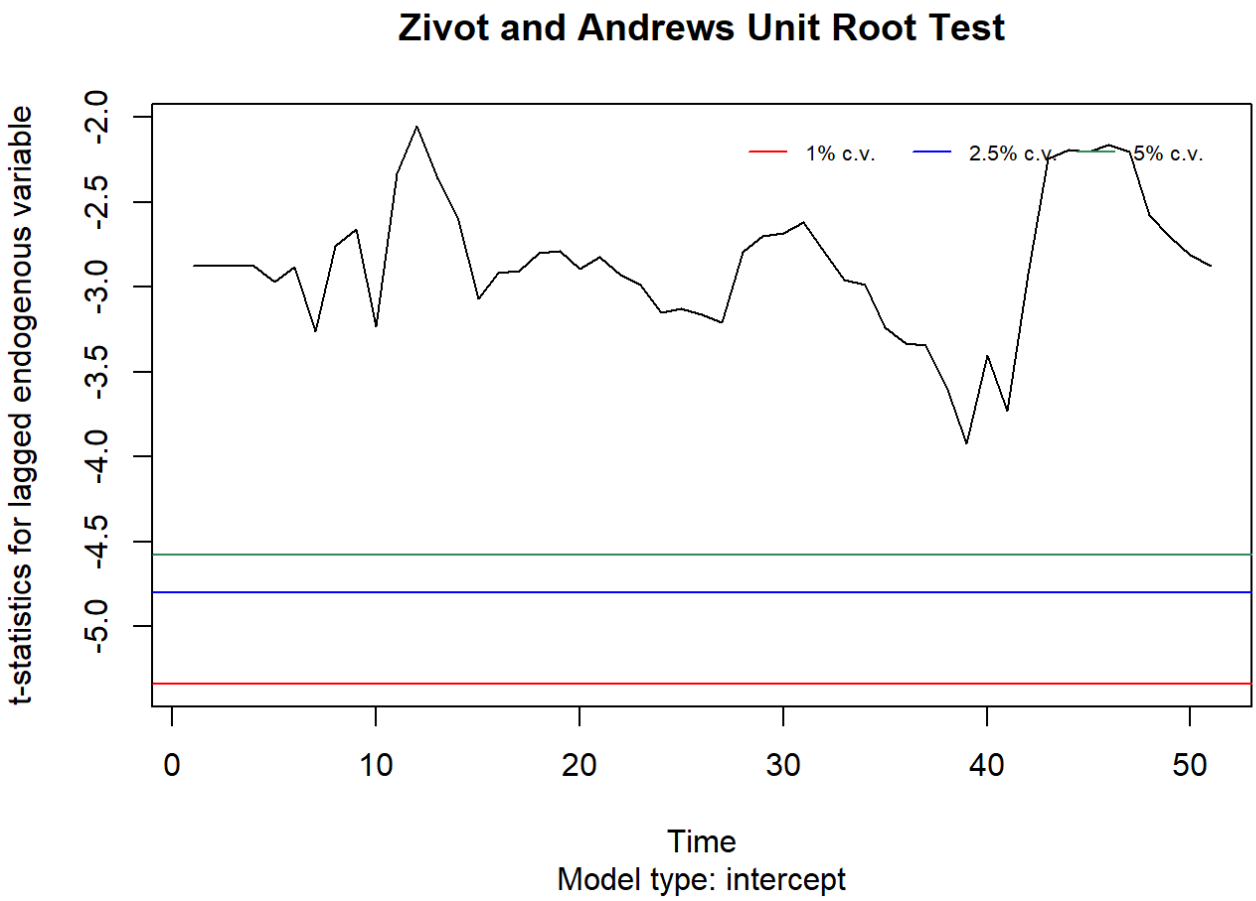
```
##
## ################################
## # Zivot-Andrews Unit Root Test #
## ################################
##
##
## Call:
## lm(formula = testmat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4397 -0.4716  0.0070  0.4731  3.1235
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.21163    3.26446   3.434  0.00140 **
## y.l1         0.63342    0.10792   5.869 7.21e-07 ***
## trend       -0.07376    0.02459  -3.000  0.00463 **
## y.dl1        0.62882    0.13934   4.513 5.51e-05 ***
## y.dl2       -0.07162    0.14851  -0.482  0.63226
## y.dl3        0.02850    0.14634   0.195  0.84659
## du          -2.89588    0.95641  -3.028  0.00430 **
## dt           0.24386    0.12334   1.977  0.05494 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.147 on 40 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.9511, Adjusted R-squared:  0.9425
## F-statistic: 111.2 on 7 and 40 DF,  p-value: < 2.2e-16
##
##
## Teststatistic: -3.3969
## Critical values: 0.01= -5.57 0.05= -5.08 0.1= -4.82
##
## Potential break point at position: 41
```

δ2 n’est pas significatif 0.05494 > 0.05, alors on emploie la spécification ”intercept”.

```
summary(ur.za(PRT_inv, model="intercept", lag=3))
```

```
##
## ################################
## # Zivot-Andrews Unit Root Test #
## ################################
##
##
## Call:
## lm(formula = testmat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3500 -0.5996 -0.1000  0.5721  3.1211
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.672578   3.039624   3.840 0.000418 ***
## y.l1         0.609164   0.099639   6.114 2.98e-07 ***
## trend       -0.062517   0.025144  -2.486 0.017070 *
## y.dl1        0.672550   0.133493   5.038 9.92e-06 ***
## y.dl2       -0.005502   0.152146  -0.036 0.971326
## y.dl3        0.074109   0.140689   0.527 0.601203
## du          -1.837601   0.742232  -2.476 0.017516 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.173 on 41 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.9476, Adjusted R-squared:   0.94
## F-statistic: 123.7 on 6 and 41 DF,  p-value: < 2.2e-16
##
##
## Teststatistic: -3.9225
## Critical values: 0.01= -5.34 0.05= -4.8 0.1= -4.58
##
## Potential break point at position: 39
```

```
plot(ur.za(PRT_inv, model="intercept", lag=3))
```



Pour la spécification" both", on a trouvé que δ2 n'est pas significatif 0.05494 > 0.05, alors on a employé la spécification "intercept". On trouve que δ1 a une p-value de 0.017516 < 0.05, donc la spécification "intercept" est correcte. La statistique calculée -3.9225 > la valeur critique à 5% =-4.8, donc on accepte H0. On a DS sans changement structurel.

Test de Lee et Strazicich

On va utiliser le test de Lee et Strazicich parce qu'il a la possibilité de deux dates de rupture, ce qui n'est pas le cas pour le test de Zivot et Andrews.

Comme on avait seulement δ1 significatif dans ZA, alors on choisit la spécification "crash".

```
source("C:\\Users\\moude\\Desktop\\M1\\S2\\Econométrie des séries temporelles\\LeeStrazicichUnitRoot-master\\LeeStrazicichUnitRoot-master\\LeeStrazicichUnitRootTestParallelization.R")
source("C:\\Users\\moude\\Desktop\\M1\\S2\\Econométrie des séries temporelles\\LeeStrazicichUnitRoot-master\\LeeStrazicichUnitRoot-master\\LeeStrazicichUnitRootTest.R")
```

Le test avec un break.

```
myBreaks <- 1
myModel <- "crash"
myLags <- 3
myLS_test <- ur.ls(y= PRT_inv , model = myModel, breaks = myBreaks, lags = myLags, method = "GTOS",pn = 0.1, print.results = "print" )
```

```
## [1] -2.850924
## [1] "First possible structural break at position: 7"
## [1] "The location of the first break - lambda_1: 0.1 , with the number of total
observations: 52"
## Critical values - Crash model:
##          1%     5%    10%
## [1,] -4.239 -3.566 -3.211
## [1] "Number of lags determined by general-to-specific lag selection: 1"
## Runtime:
## Time difference of 0.0006307999 mins
```

Comme La valeur de la t statistique calculée est -2.850924 > -3.566, on accepte H0 donc le PGD est DS sans changement structurel.

Le test avec deux breaks.

```
myBreaks <- 2
myModel <- "crash"
myLags <- 3
myLS_test <- ur.ls(y= PRT_inv , model = myModel, breaks = myBreaks, lags = myLags,
method = "GTOS",pn = 0.1, print.results = "print" )
```

```
## [1] -3.077443
## [1] "First possible structural break at position: 7"
## [1] "The location of the first break - lambda_1: 0.1 , with the number of total
observations: 52"
## [1] "Second possible structural break at position: 10"
## [1] "The location of the second break - lambda_2: 0.2 , with the number of tota
l observations: 52"
## Critical values:
##               Break 2 - 0.4 - 1% Break 2 - 0.4 - 5% Break 2 - 0.4 - 10%
## Break 1 - 0.2              -6.16              -5.59               -5.27
## Break 1 - 0.4                 NA                 NA                  NA
## Break 1 - 0.6                 NA                 NA                  NA
##               Break 2 - 0.6 - 1% Break 2 - 0.6 - 5% Break 2 - 0.6 - 10%
## Break 1 - 0.2              -6.41              -5.74               -5.32
## Break 1 - 0.4              -6.45              -5.67               -5.31
## Break 1 - 0.6                 NA                 NA                  NA
##               Break 2 - 0.8 - 1% Break 2 - 0.8 - 5% Break 2 - 0.8 - 10%
## Break 1 - 0.2              -6.33              -5.71               -5.33
## Break 1 - 0.4              -6.42              -5.65               -5.32
## Break 1 - 0.6              -6.32              -5.73               -5.32
## [1] "Number of lags determined by general-to-specific lag selection: 1"
## Runtime:
## Time difference of 0.01258752 mins
```

La valeur de la t statistique calculée est -3.077443 > -5.59, on accepte H0 donc le PGD est DS sans changement structurel.

Bootstrap Lee et Strazicich

Vue qu'on a un petit échantillon de 52 observations, on va utiliser ls.bootstrap.

Bootstrap un break.

```
cl <- makeCluster(max(1, detectCores() - 1))
registerDoSNOW(cl)
myBreaks <- 1
myModel <- "crash"
myParallel_LS <- ur.ls.bootstrap(y = PRT_inv , model = myModel, breaks = myBreaks,
lags = myLags, method = "Fixed",pn = 0.1, critval = "bootstrap", print.results =
"print")
```

```
## [[1]]
## [1] -1.928239
##
## [1] "First possible structural break at position: 15"
## [1] "The location of the first break - lambda_1: 0.3 , with the number of total
observations: 52"
## Critical values - Crash model:
##          1%     5%    10%
## [1,] -4.239 -3.566 -3.211
## [1] "Number of lags used: 3"
## Runtime:
## Time difference of 0.004070683 mins
```

λ est estimé à 0.3 et Tb = 15 donc 1984 est la date de rupture. Comme La valeur de la t statistique calculée est -1.928239 > -3.566, on accepte H0 donc le PGD est DS sans changement structurel.

Bootstrap deux break.

```
cl <- makeCluster(max(1, detectCores() - 1))
registerDoSNOW(cl)
myBreaks <- 2
myModel <- "crash"
myParallel_LS <- ur.ls.bootstrap(y = PRT_inv , model = myModel, breaks = myBreaks,
lags = myLags, method = "Fixed",pn = 0.1, critval = "bootstrap", print.results =
"print")
```

```
## [[1]]
## [1] -2.20179
##
## [1] "First possible structural break at position: 15"
## [1] "The location of the first break - lambda_1: 0.3 , with the number of total
observations: 52"
## [1] "Second possible structural break at position: 17"
## [1] "The location of the second break - lambda_2: 0.3 , with the number of tota
l observations: 52"
## Critical values:
##               Break 2 - 0.4 - 1% Break 2 - 0.4 - 5% Break 2 - 0.4 - 10%
## Break 1 - 0.2              -6.16              -5.59               -5.27
## Break 1 - 0.4                 NA                 NA                  NA
## Break 1 - 0.6                 NA                 NA                  NA
##               Break 2 - 0.6 - 1% Break 2 - 0.6 - 5% Break 2 - 0.6 - 10%
## Break 1 - 0.2              -6.41              -5.74               -5.32
## Break 1 - 0.4              -6.45              -5.67               -5.31
## Break 1 - 0.6                 NA                 NA                  NA
##               Break 2 - 0.8 - 1% Break 2 - 0.8 - 5% Break 2 - 0.8 - 10%
## Break 1 - 0.2              -6.33              -5.71               -5.33
```

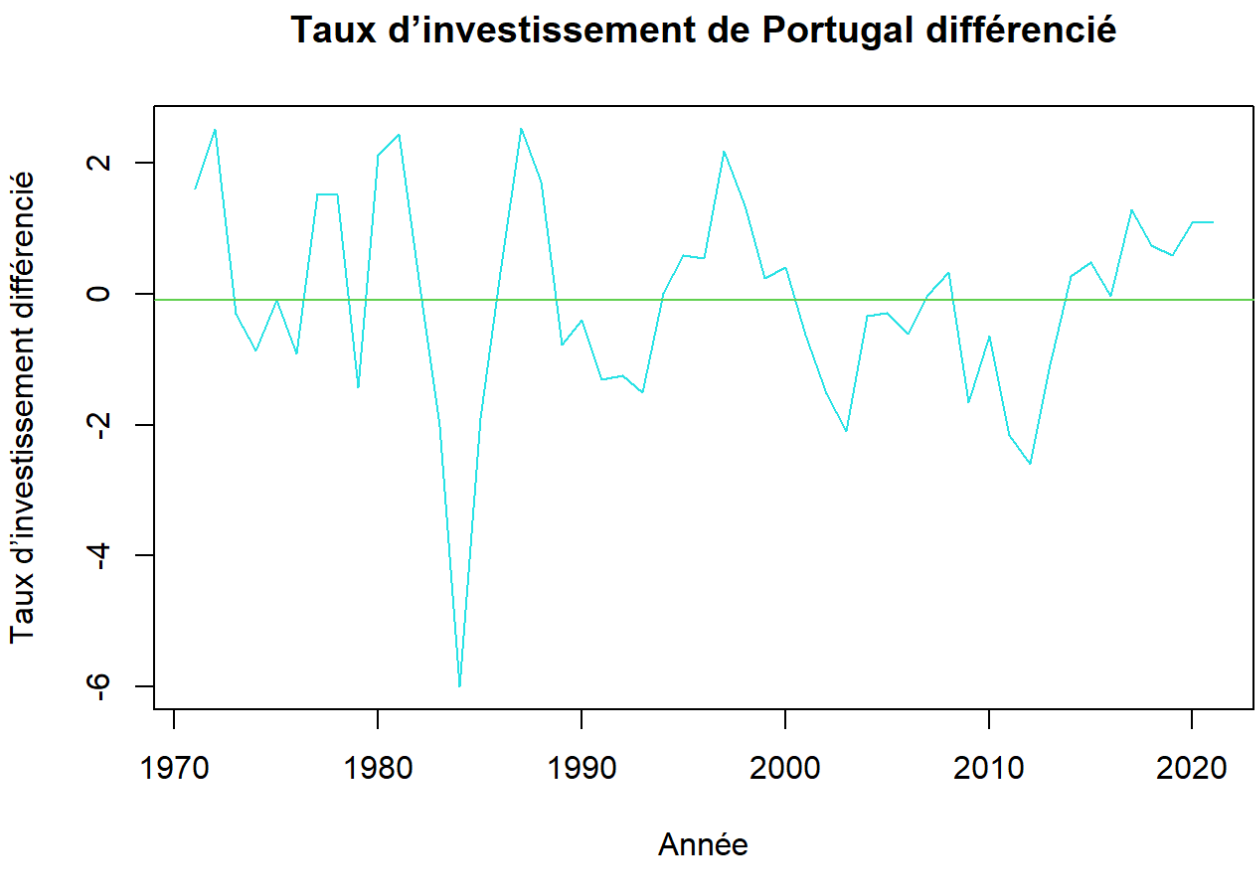
```
## Break 1 - 0.4              -6.42              -5.65              -5.32
## Break 1 - 0.6              -6.32              -5.73              -5.32
## [1] "Number of lags used: 3"
## Runtime:
## Time difference of 0.01203145 mins
```

La valeur de la t statistique calculée est -2.20179 > -5.59, on accepte H0 donc le PGD est DS sans changement structurel.

Differentiation

On a un processus DS qui s'écrit comme : Xt = δ + Xt−1 + ut Avec ut un BB et δ une constante appelée la dérive.

```
D_PRT_inv = diff(PRT_inv)
moy = mean(D_PRT_inv)
plot.ts(D_PRT_inv,xlab="Année", ylab="Taux d'investissement différencié", main="Ta
ux d'investissement de Portugal différencié",type='l',col=5)
abline(h = moy,col = 3)
```



La série différenciée presente des proprietes de stationnarité, comme la variation autour de la moyenne, dans ce cas 0, et une variance plus ou moins constante. Donc, on peut passer à la modélisation.

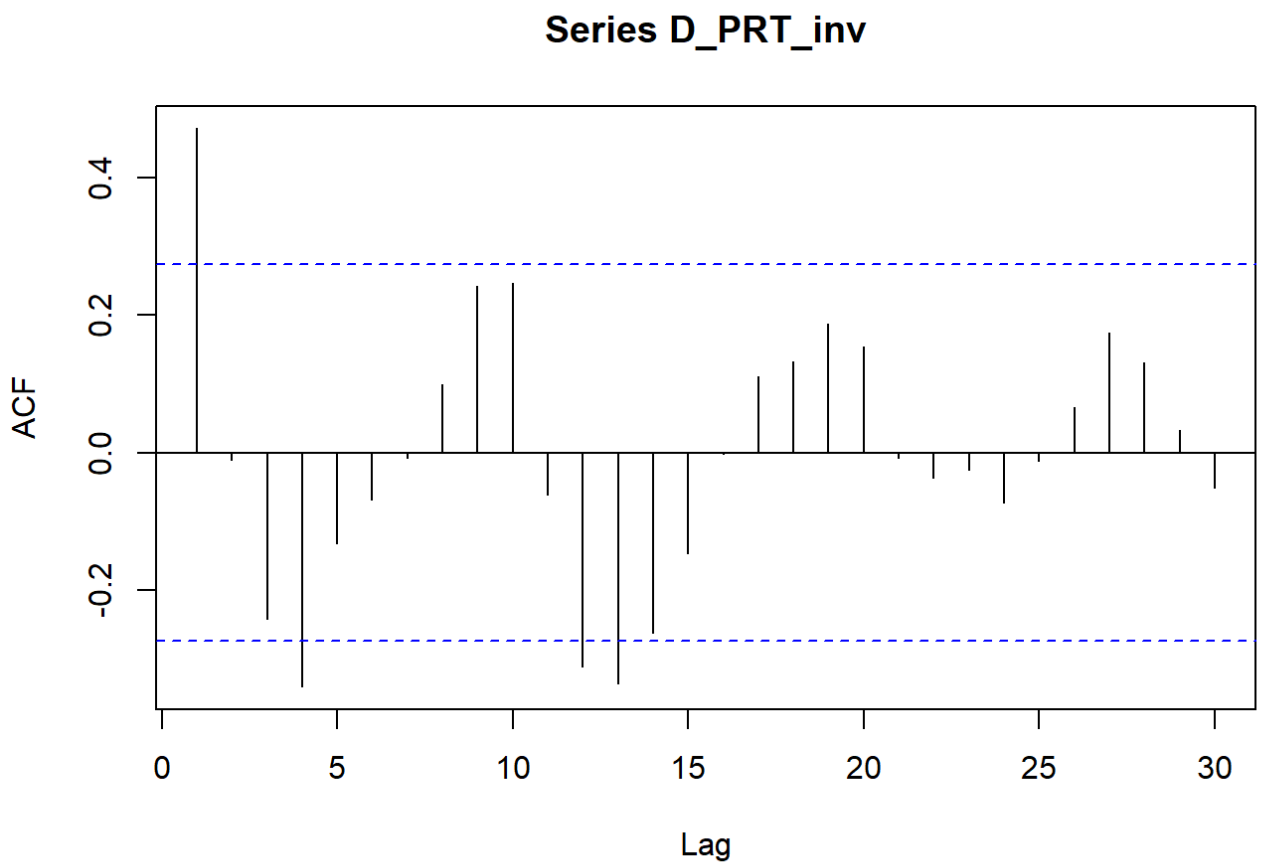
Modélisation

Maintenant qu'on a une série stationnaire on peut passer à la modélisation. En premier on affiche les graphes d'ACF et de PACF pour voir s'il y a de l'autocorrélation et un test de Ljung-box pour confirmer qu'il

y a de l'autocorrélation et puis on utilise la fonction eacf() pour obtenir les valeurs de p et q du modèle et on va simuler plusieurs modèles afin de choisir le meilleur.

Plot ACF

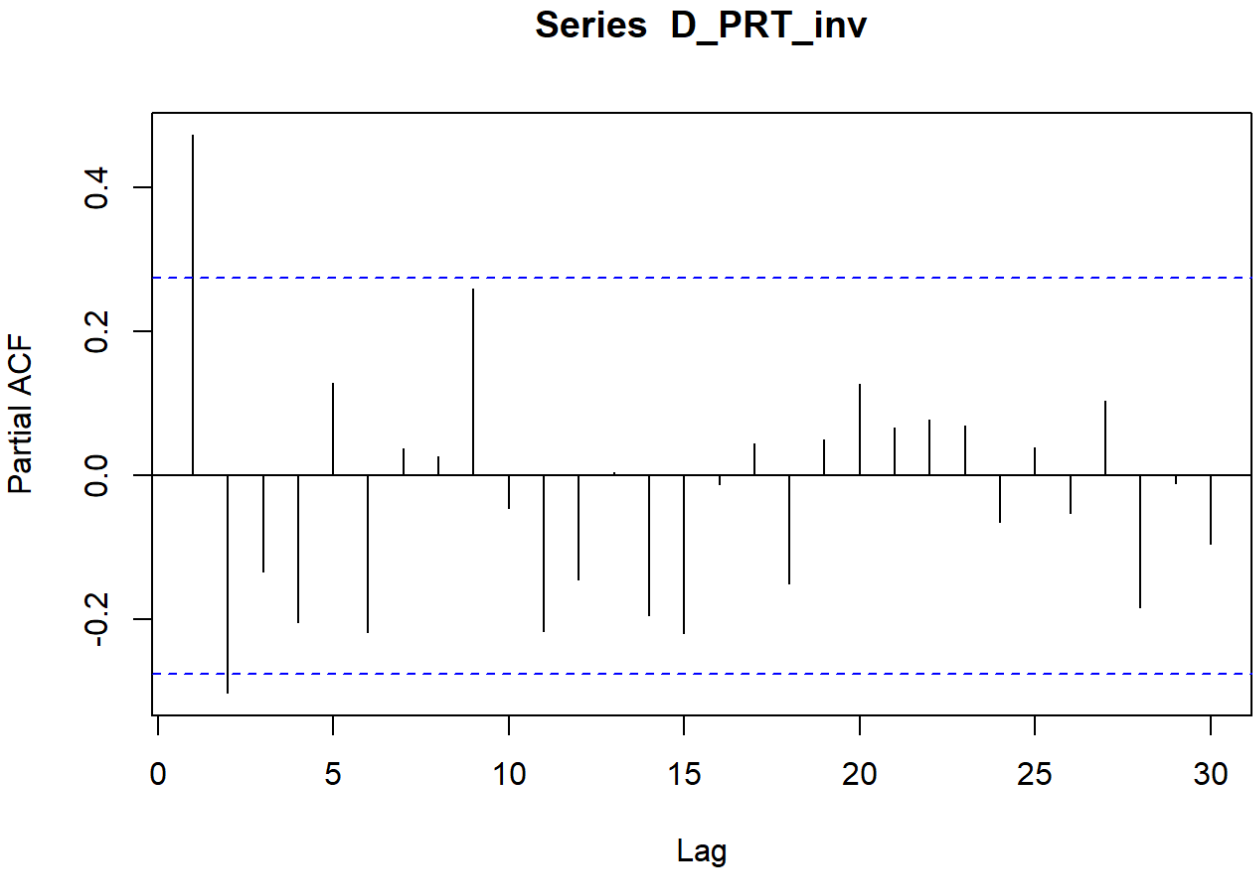
```
acf(D_PRT_inv, lag.max = 30)
```



On peut voir qu'il y a de l'autocorrélation jusqu'à l'ordre 13. Les lags significatifs sont 1, 4, 12 et 13. On constate aussi que les coefficients de corrélation se dégradent progressivement, on peut dire qu'il s'agit probablement d'un processus AR.

Plot PACF

```
pacf(D_PRT_inv, lag.max = 30)
```



Pour la PACF il y a de l'autocorrélation jusqu'à l'ordre 2. Il y a que deux lags significatifs 1 et 2.

On fait un test de Ljung-box pour confirmer l'existence d'autocorrélation.

Test d'autocorrelation

```
Box.test(D_PRT_inv, type = "Ljung-Box", lag = 13)
```

```
##
##  Box-Ljung test
##
## data:  D_PRT_inv
## X-squared = 46.836, df = 13, p-value = 1.03e-05
```

La p-value est inférieure à 0.05, on rejette H0, ce qui confirme qui st qu'il y a de l'autocorrélation.

EACF

```
eacf(D_PRT_inv)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o o x o o o o o o o  o  x  o
## 1 x o o x o o o o o o o  o  o  o
## 2 x o o x o o o o o o o  o  o  o
## 3 x o o o o o o o o o o  o  o  o
## 4 x o o o o o o o o o o  o  o  o
## 5 x x x o o o o o o o o  o  o  o
```

```
## 6 o o x o o o o o o o o  o  o  o
## 7 x o x o o o o x o o o  o  o  o
```

On obtient les modeles suivants : ARMA (0,13), ARMA (1,4), ARMA (2, 4), ARMA (7,8).

ARMA (0,13)

```
reg1 = Arima(D_PRT_inv, order = c(0,0,13))
coeftest(reg1)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value Pr(>|z|)
## ma1         0.3705963  0.1788485  2.0721  0.03825 *
## ma2        -0.1314625  0.2902675 -0.4529  0.65062
## ma3        -0.2356662  0.2193123 -1.0746  0.28257
## ma4        -0.4467875  0.2846005 -1.5699  0.11644
## ma5         0.0774404  0.2558679  0.3027  0.76215
## ma6         0.1258501  0.2603462  0.4834  0.62882
## ma7         0.2544566  0.2797785  0.9095  0.36309
## ma8         0.1775511  0.2131111  0.8331  0.40477
## ma9        -0.1880127  0.2419218 -0.7772  0.43706
## ma10       -0.0062261  0.2439100 -0.0255  0.97964
## ma11        0.0172954  0.1991164  0.0869  0.93078
## ma12       -0.5638212  0.2311975 -2.4387  0.01474 *
## ma13       -0.4511873  0.1961040 -2.3008  0.02141 *
## intercept -0.1518462  0.0829022 -1.8316  0.06701 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# On enleve les coefficient non significatifs
reg1_x = Arima(D_PRT_inv, order = c(0,0,13),fixed=c(NA, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0))
coeftest(reg1_x)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1  0.64990    0.14077  4.6169 3.895e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On arrive au modèle suivant après qu'on enlève tous les coefficients non significatifs : Yt = θ1ϵt−1 + ϵt

ARMA (1,4)

```
reg2 = Arima(D_PRT_inv, order = c(1,0,4))
coeftest(reg2)
```

```
##
## z test of coefficients:
##
```

```
##            Estimate Std. Error z value  Pr(>|z|)
## ar1         0.434093   0.177697  2.4429 0.0145704 *
## ma1         0.080959   0.147972  0.5471 0.5842930
## ma2        -0.258307   0.183462 -1.4080 0.1591441
## ma3        -0.226403   0.118006 -1.9186 0.0550374 .
## ma4        -0.596236   0.208828 -2.8552 0.0043016 **
## intercept -0.205439   0.058726 -3.4983 0.0004683 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# On enleve les coefficient non significatifs
reg2_x = Arima(D_PRT_inv, order = c(1,0,4),fixed=c(NA, 0, 0, NA, NA, NA))
coeftest(reg2_x)
```

```
##
## z test of coefficients:
##
##            Estimate Std. Error z value Pr(>|z|)
## ar1         0.424775   0.139726  3.0401 0.002365 **
## ma3        -0.270168   0.113719 -2.3758 0.017513 *
## ma4        -0.637631   0.133544 -4.7747  1.8e-06 ***
## intercept -0.200712   0.067108 -2.9909 0.002782 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On arrive au modèle suivant après qu'on enlève tous les coefficients non significatifs : $Yt = c + \varphi 1Xt-1 + \theta 3\epsilon t-3 + \theta 4\epsilon t-4 + \epsilon t$

ARMA (2,4)

```
reg3 = Arima(D_PRT_inv, order = c(2,0,4))
coeftest(reg3)
```

```
##
## z test of coefficients:
##
##            Estimate Std. Error z value  Pr(>|z|)
## ar1         0.493967   0.215592  2.2912 0.0219514 *
## ar2        -0.130537   0.217863 -0.5992 0.5490608
## ma1         0.027646   0.184001  0.1502 0.8805681
## ma2        -0.180690   0.191154 -0.9453 0.3445264
## ma3        -0.204423   0.128172 -1.5949 0.1107316
## ma4        -0.642359   0.190997 -3.3632 0.0007705 ***
## intercept -0.212585   0.054067 -3.9319 8.427e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# On enleve les coefficient non significatifs
reg3_x = Arima(D_PRT_inv, order = c(2,0,4),fixed=c(NA, 0, 0, 0, NA, NA, NA))
coeftest(reg3_x)
```

```
##
## z test of coefficients:
```

```
##
##           Estimate Std. Error z value  Pr(>|z|)
## ar1        0.42478    0.13978  3.0388  0.002375 **
## ma3       -0.27016    0.11372 -2.3757  0.017516 *
## ma4       -0.63762    0.13355 -4.7745 1.801e-06 ***
## intercept -0.20071    0.06711 -2.9908  0.002783 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On arrive au modèle suivant après qu'on enlève tous les coefficients non significatifs : Yt = c + φ1Xt−1 + θ3ϵt−3 + θ4ϵt−4 + ϵt

ARMA (7,8)

```
reg4 = Arima(D_PRT_inv, order = c(7,0,8))
coeftest(reg4)
```

```
##
## z test of coefficients:
##
##           Estimate Std. Error z value Pr(>|z|)
## ar1        0.60141    0.19017  3.1625 0.001564 **
## ar2        0.15799    0.23738  0.6656 0.505698
## ar3        0.33684    0.25507  1.3206 0.186638
## ar4       -0.72494    0.17913 -4.0469 5.19e-05 ***
## ar5        0.26709    0.22785  1.1722 0.241113
## ar6       -0.35323    0.25420 -1.3896 0.164665
## ar7        0.43157    0.20504  2.1048 0.035311 *
## ma1       -0.12952    0.21577 -0.6003 0.548314
## ma2       -0.51855    0.48754 -1.0636 0.287509
## ma3       -0.64893    0.34606 -1.8752 0.060771 .
## ma4        0.14876    0.68597  0.2169 0.828312
## ma5        0.45325    0.26750  1.6944 0.090194 .
## ma6        0.72686    0.75392  0.9641 0.334991
## ma7       -0.20016    0.22607 -0.8854 0.375925
## ma8       -0.83171    0.43399 -1.9165 0.055307 .
## intercept -0.14719    0.10386 -1.4172 0.156437
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# On enleve les coefficient non significatifs
reg4_x = Arima(D_PRT_inv, order = c(7,0,8),fixed=c(NA, 0, 0, 0, 0, 0, 0, 0, NA, 0,
0, 0, 0, 0, 0), include.mean= FALSE)
coeftest(reg4_x)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ar1  0.61786    0.12708  4.8618 1.163e-06 ***
## ma2 -0.54540    0.18834 -2.8958  0.003782 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On arrive au modèle suivant après qu'on enlève tous les coefficients non significatifs : $Yt = \varphi 1Xt-1 + \theta 2\epsilon t-2 + \epsilon t$

On peut voir aussi le modèle proposé par R.

```
reg_auto = auto.arima(D_PRT_inv)
coeftest(reg_auto)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1  0.64990    0.14077  4.6169 3.895e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

R propose un modèle ARIMA(0,0,1).

Le BIC de chaque modèle.

```
BIC(reg1_x)
```

```
## [1] 180.6155
```

```
BIC(reg2_x)
```

```
## [1] 184.5212
```

```
BIC(reg3_x)
```

```
## [1] 184.5212
```

```
BIC(reg4_x)
```

```
## [1] 183.8448
```

```
BIC(reg_auto)
```

```
## [1] 180.6155
```

On choisit celui avec le plus petit BIC, dans ce cas, le modèle 1 ARIMA(0,0,1)

Tests d'autocorrélation sur les résidus

T test

```
t.test(reg1_x$residuals)
```

```
##
##  One Sample t-test
##
## data:  reg1_x$residuals
## t = -0.27253, df = 50, p-value = 0.7863
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.4219895  0.3211567
## sample estimates:
##   mean of x
## -0.05041642
```

La p-value (0.7863) est supérieure à 5%, on accepte H0, donc l'espérance des aléas est nulle.

Archtest

```
ArchTest(reg1_x$residuals, lags = 13)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  reg1_x$residuals
## Chi-squared = 6.8105, df = 13, p-value = 0.9117
```

La p-value (0.9117) est supérieure à 5%, on accepte H0, donc il n'y a pas d'effet d'Arch.

Ljung-Box test

```
Box.test(reg1_x$residuals, type="Ljung-Box", lag=13)
```

```
##
##  Box-Ljung test
##
## data:  reg1_x$residuals
## X-squared = 17.126, df = 13, p-value = 0.1936
```

La p-value (0.1936) est supérieure à 5%, on accepte H0, donc il n'y a pas d'autocorrélation.

Prévision

Sur la série différencié.

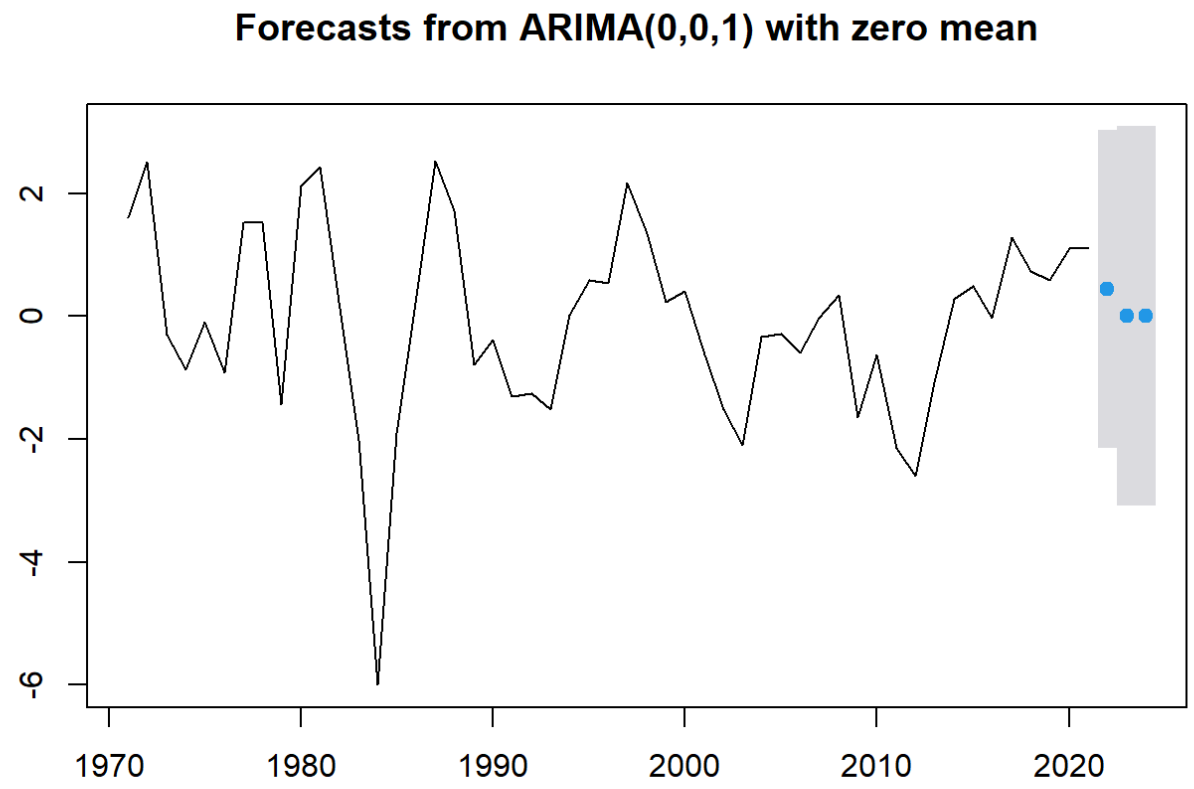
```
reg_d = Arima(D_PRT_inv, order = c(0,0,1), include.mean= FALSE)
coeftest(reg_d)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1  0.64990    0.14077  4.6169 3.895e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
prev <- forecast(reg_d, h = 3, level = 0.95)
prev
```

```
##      Point Forecast     Lo 95    Hi 95
## 2022      0.4444667 -2.146818 3.035751
## 2023      0.0000000 -3.090454 3.090454
## 2024      0.0000000 -3.090454 3.090454
```

```
plot(prev)
```



Sur la série originale.

```
reg_n = Arima(PRT_inv, order = c(0,1,1), include.mean= FALSE)
coeftest(reg_n)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1  0.64991    0.14077  4.6169 3.895e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
prev <- forecast(reg_n, h = 3, level = 0.95)
prev
```

```
##      Point Forecast    Lo 95    Hi 95
## 2022       20.76447 18.17317 23.35576
## 2023       20.76447 15.76509 25.76384
## 2024       20.76447 14.18627 27.34267
```

```
plot(prev)
```